APPLICATION UNDER UNITED STATES PATENT LAWS

Atty. Dkt. No. PW 281571
(M#)

Invention: MANAGED OBJECT REPLICATION AND DELIVERY

Inventor (s): Steven L. SEED
Kevin HOBBS
Shane M. GLYNN
Isaac W. FORAKER
Peter J. JONES
Homer H. CHEN

Pillsbury Winthrop LLP

This is a:

☐ Provisional Application

☒ Regular Utility Application

☐ Continuing Application
☒ The contents of the parent are incorporated by reference

☐ PCT National Phase Application

☐ Design Application

☐ Reissue Application

☐ Plant Application

☐ Substitute Specification
Sub. Spec Filed ________
in App. No. ___/________

☐ Marked up Specification re
Sub. Spec. filed ________
In App. No ___/________

SPECIFICATION

MANAGED OBJECT REPLICATION AND DELIVERY

BACKGROUND

[0001] This invention relates in general to the field of computer networks. Particularly, aspects of this invention pertain to managed object replication and delivery over a network.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Exemplary embodiments of the invention are illustrated in the accompanying drawings in which like references indicate similar or corresponding elements and in which:

[0003] FIG. 1 is a high-level block diagram of a topology of the managed object replication and delivery method and system according to embodiments of the invention;

[0004] FIG. 2 is a high-level block diagram illustrating the data flows of managed object replication and delivery method according to embodiments of the invention;

[0005] FIGS. 3(a), 3(b) and 3(c) are a flow chart of the managed object replication and delivery method and the object purging method according to embodiments of the invention;

[0006] FIG. 4 is a flow chart of a popularity computation according to embodiments of the invention;

[0007] FIG. 5 is a flow chart of a replication scheme according to embodiments of the invention;

[0008] FIG. 6 is a flow chart of a purge scheme according to embodiments of the invention; and

[0009] FIG. 7 is a block diagram of the managed object replication and delivery system according to embodiments of the invention.

DETAILED DESCRIPTION

[0010] A typical content delivery network (CDN) operator deploys one or more parent servers, hosting a plurality of objects, in a network and one or more edge servers at the edge of the network to facilitate more cost-effective and efficient delivery of such objects to an end-user (client). End-users or client proxies that access customers' objects are called clients. Content provider companies, organizations, etc. that subscribe to the CDN service are referred to as customers. As used herein, an object includes, without limitation, an audio file (such as, e.g., an MP3 (Motion Picture Experts Group-1 Layer 3) file and a RealNetworks, Inc. Real format file), a video file (such as an MPEG file), an image file (such as, e.g., a BMP (bitmap) file or JPEG (Joint Photographic Experts) file) and any other software or data file or object. It is typically desirable to serve objects from edge servers because the edge servers are typically closer (by various measures of distance) to end-users. For example, streaming content data from edge servers saves parent-to-edge bandwidth. Furthermore, the less the distance objects must travel can also mean reduced network congestion and packet losses, which can lead to a better experience for the end-user through faster response times and better quality of service.

[0011] It is typically not feasible to store all objects on the edge servers. The main difficulty is due to the fact that many such objects are very large (typically on the order of

10 MB (10,000,000 bytes) - - in the neighborhood of 500 MB for movies). The storage and rack space required to accommodate often large and sometimes rarely requested objects at every edge server can be cost prohibitive as the number of customers grows and the number of their objects increases. It may not even be possible to store a good working set of objects, for example a set of objects thought to be requested often and/or better suited to be served from an edge server, because of the size and changing demand for objects in the working set.

[0012] One obvious solution is to pre-populate edge servers with objects for which there will likely be a significant or high demand. However, it is difficult to predict popularity and difficult to manage pre-populating. A related solution is to associate objects with two or more domains depending on popularity of the object, e.g., one domain for popular objects (served from edge servers) and another domain for less popular objects (served from parent servers). However, this requires some way to pre-determine what objects are popular and what objects are less popular statically, and build that popularity into the domain name of the object. As with pre-populating, it is difficult to predict popularity and to manage assignment of domains based on such popularity determinations.

[0013] Other solutions fetch objects on demand. In such schemes, when a requested object is not available on a handling edge server, a connection is made between a parent server having the requested object and the handling edge server to fetch the requested object from the parent server. Such fetching suffers however from having to go through the parent path (the network path between the handling edge server and the parent server with the object) whenever a client requests an object that is not already at the particular edge server.

[0014] Fetching a large object to the handling edge server through a parent path can be slow. For example, there may be limited available bandwidth from the parent server to the handling edge server, i.e., sometimes the parent path has less bandwidth than even the network path from the edge server to the client (e.g., the "last mile" in a broadband network). If a parent server uses too much bandwidth copying an object to an edge server, this can create congestion at that parent server. If storage fill bandwidth is matched to client bandwidth, it is difficult to handle a second, faster client and if fetch is done using a streaming protocol (for instance, the Real-Time Streaming Protocol (RTSP) and Real-Time Transport Protocol (RTP) standards), the quality of the copy made can be hurt due to lost packets ("thinning").

[0015] Moreover, there may be an unreliable end-to-end parent path due to network congestion. And, if a parent server has to preprocess an object (e.g., to generate an image at a specific bit rate) or is otherwise busy with other tasks, this may further slow its ability to serve the request for the object fast enough. For example, if a client requests a bit rate higher than the parent-to-edge bit rate, delays will likely occur. Under such conditions, the parent server may fail, for example, to stream the object in time or to maintain the stream of an object at a requested bit rate thereby causing a thinned object, i.e., an object with lower quality due to lost packets in its transmission, to be populated at the edge server and delivered to subsequent clients requesting the same object.

[0016] Thus, it would be advantageous to populate edge servers with the most popular objects yet somehow serve the rest from parent servers with a goal to maximize the amount of object bits served from edge servers of the network. It would also be advantageous to populate edge servers by, for example, storage fill on demand when an object is popular enough, without having to make the end-user wait for such population. Therefore, it would be advantageous to provide a method and system for managed object replication and delivery over a network.

[0017] According to embodiments of the invention, a method and system for managed object replication and delivery over a network redirects, directly or indirectly, a client's request for an object that is not available at a best or optimal handling edge server of the network to a parent server of the network that has the requested object. So, where the requested object is not available at the handling edge server, the client's request is redirected directly to the parent server that can provide the requested object to the client or indirectly via one or more parent servers to a parent server that can provide the requested object to the client. The method and system further intelligently replicates the object to the edge server if the object is popular enough. Likewise, an object is removed from an edge server when the object is no longer popular. All redirection and replication operations are preferably transparent to the end-user and do not degrade the quality of service. Other embodiments of the invention are possible and some are described hereafter.

[0018] So, for example, under the framework described herein, a request for a streaming object will be served by a handling edge server if that handling edge server has a copy of that object. Otherwise, the request is redirected, directly or indirectly, to a parent server for service of the requested streaming object to the client. If the requested streaming object is popular, the object is replicated from a parent server that has the requested streaming object to the handling edge server so that the handling edge server will serve the object from the edge of the network when the object is requested in the future. If a streaming object is no longer popular, the object is removed from an edge server.

[0019] As used herein, replication generally refers to the permanent and/or volatile storage of an object in a server, particularly an edge server and if applicable, a parent server. Accordingly, the term replication will be considered synonymous to storing, caching and copying. In typical embodiments, replication of an object will usually refer to temporary storage of the object in an edge server and/or a parent server for an undefined duration.

[0020] A typical network for the managed object replication and delivery method according to embodiments of the invention is illustrated in FIG. 1. The network 100 comprises one or more parent server sites 120 and one or more edge server sites 130. The network also optionally has access to one or more origin server sites 110. The origin server sites are typically owned and/or maintained by the network provider's customers for storing and serving one or more objects. Each customer (content provider) may have its own origin server site. Furthermore, one or more clients 140 access the network to request one or more objects. A parent server site (or simply parent site or parent server) may comprise one parent server or a cluster of parent servers. Likewise, an edge server site (or simply edge site or edge server) may comprise one edge server or a cluster of edge servers and an origin server site (or simply origin site or origin server) may comprise one origin server or a cluster of origin servers. Typically, the network 100 is configured such that servers in a cluster share a common storage. In any event, configuration details of the parent server site, edge server site, and the origin server site are not important to the present invention.

[0021] In the typical network, the parent servers and edge servers are maintained by a network provider, wherein the parent servers are primarily used for storing and managing one or more objects and edge servers are primarily used for serving objects to clients. In some embodiments, all the objects are retrieved from origin servers and stored over one or more parent servers before any end-users can access each such object as the object is stored on the parent servers. Accordingly, in these embodiments, the origin servers play no significant role in the managed object replication and delivery method except to supply new and/or updated objects for storage on the parent servers. Moreover, only the parent servers communicate with the origin servers. In other embodiments, each requested object is replicated from one or more origin servers to one or more parent servers (and/or one or more edge servers) when the requested object becomes popular (as described in more detail below). In these embodiments, the origin servers play a more significant role in the managed object replication and delivery method to supply objects to parent and/or edge servers when requested. So, in these embodiments, the origin servers and parent servers communicate between each other and the origin servers and clients may also communicate between each other. In all of these embodiments, the communications relationships between origin servers and parent servers may be one-to-one, one-to-many or many-to-many.

[0022] Further, as shown in FIG. 1, the parent servers and edge servers communicate between each other, edge servers and clients communicate between each other and parent servers and clients communicate between each other. While in embodiments, as shown in FIG. 1, the edge servers have a one-to-one or one-to-many communications relationship with parent servers, edge servers may also have many-to-many communications relationships with parent servers. As discussed in more detail below, the edge servers act as the primary source of serving objects but if a requested object is not available at the edge server a parent server that has the requested object will serve the requested object to the clients. Also, FIG. 1 shows a single layer or level of parent servers and origin servers. As will be apparent to those skilled in the art, more than one layer or level of parent servers and/or origin servers may be used.

[0023] According to embodiments of the invention and referring to FIGS. 2, 3(a), 3(b) and 3(c), the method of managed object replication and delivery and the method of object purging is depicted. FIG. 2 depicts embodiments of the method in relation to a portion of the network 100, an origin server 110 and a client 140 as shown in FIG. 1. FIGS. 3(a), 3(b) and 3(c) depict embodiments of the method in flowchart form.

[0024] Initially, the method of managed object replication and delivery directs (at 200, 300) a client, requesting one or more objects, to an edge server in the network, whether or not the edge server has the requested object(s). Preferably, the client is directed to an optimal edge server, e.g., based on network traffic conditions and server load. As will be apparent to those skilled in the art, any number of currently known or future developed mechanisms may be used to select a best or optimal edge server. Determination of a best or optimal edge server preferably includes selection of an edge server most suitable for delivery of one or more objects to the client according to any number of currently known or future developed algorithms. For example, determination of a best or optimal edge server may be performed based on the likelihood of a copy of the requested object(s) being available at the candidate edge server, on the bandwidth between a candidate edge server and the client, on a best repeater selector (for example, as described in U.S. Patent No. 6,185,598) and/or on any number of other criteria.

[0025] The selected best or optimal edge server 130 determines (at 305) whether the edge server already has the requested object and, if so, serves (at 205, 310) the object to the requesting client 140. For example, the selected edge server 130 will check its storage to determine whether the requested object is available and if so, may serve the object to the requesting client 140.

[0026] If the selected edge server does not have the requested object, a check is initiated (at 315) for the edge server to determine whether the requested object is popular and if so, to replicate the popular requested object to the edge server. In embodiments, the method depicted in FIG. 3(b) and discussed in more detail below is employed to determine whether the requested object is popular and if so, to replicate the popular requested object to the edge server.

[0027] In embodiments, the checking of whether the requested object is popular and replicating the popular requested object to the edge server may be performed independently of one or more functions of the method of managed object replication and delivery, such as the checking if a server has the requested object and serving the requested object to the client if the server has the requested object or redirecting the client to a server that has the requested object (and serving the requested object to the client). Thus, in embodiments, the checking of whether the requested object is popular and replicating the popular object to the edge server may be performed in parallel with or before the performance of certain functions of the method of managed object replication and delivery such as the checking if a server has the requested object and serving the requested object to the client if the server has the requested object or redirecting the client to a server that has the requested object (and serving the requested object to the client). Advantageously, should the checking, redirecting and serving of the requested object fail, the checking of whether the requested object is popular and replicating the popular object to the edge server can manage the continued delivery of objects to clients from edge servers. Similarly, if the checking of whether the requested object is popular and replicating the popular object to the edge server should fail, the checking, redirecting and serving of the requested object can manage the continued delivery of objects from servers in the network.

[0028] Further, if the selected edge server does not have the requested object, the selected edge server directs (at 210, 320) the requesting client 140 to a parent server 120. Preferably the client 140 is redirected to a parent server that has the requested object and is able to serve (at 215, 345) the requested object to the client. If a parent server does not have (at 325) the requested object, a check is initiated (at 330) for the parent server to determine whether the requested object is popular and if so, to replicate the popular requested object to the parent server. In embodiments, the method depicted in FIG. 3(b) and discussed in more detail below is employed to determine whether the requested object is popular and if so, to replicate the popular requested object to the parent server. As with the check for the edge server, in embodiments, the checking of whether the requested object is popular and replicating the popular requested object to the parent server is performed independently of one or more functions of the method of managed object replication and delivery such as the checking if a server has the requested object and serving the requested object to the client if the server has the requested object or redirecting the client to a server that has the requested object (and serving the requested object to the client). Thus, in embodiments, the checking of whether the requested object is popular and replicating the popular requested object to the parent server may be performed in parallel with or before one or more functions of the method of managed object replication and delivery such as the checking if a server has the requested object and serving the requested object to the client if the server has the requested object or redirecting the client to a server that has the requested object (and serving the requested object to the client).

[0029] Further, if a parent server does not have the requested object, the parent server could itself use a redirection technique recursively (at 325, 335, 320) until a final parent server is reached that has the requested object. The parent server that has the requested object serves (at 215, 345) the object to the client. If the object is determined to be unavailable (at 335) (from all parent servers), an error message is returned (at 340) regarding the unavailability of the requested object.

[0030] As will be apparent to those skilled in the art, numerous methods are available to redirect a requesting client to another parent server, depending on the protocol(s) used to request the object. A handling edge server may request information from a database about to which parent server the client should be redirected. In an implementation, the edge server might have a local database, populated by pushes of redirection data from one or more servers in the network. The edge server may also simply query one or more servers in the network to identify one or more parent servers to which the client can be directed. When more than one parent server responds, the edge server may redirect the client to the parent server that responds to the query first, the edge server may redirect the client to the parent server that is topologically closest to the edge server in the network or the edge server may redirect the client to the parent server that represents the best or optimal candidate based on criteria such as network efficiency, bandwidth requirement and/or cost. Alternatively, an edge server may always go to default parent servers. Or, as discussed in relation to edge servers, a best or optimal parent server may be determined using any of the techniques outlined above. Redirection may be performed by simply sending the request onto a parent server or returning redirection information to the client for accessing the parent server. As will be apparent to those skilled in the art, any number of implementations may be used to provide the redirection information to the handling edge server.

[0031] In other embodiments, where the parent servers collectively are not populated with all of the objects and the network has access to the origin server of a requested object, the client may be redirected (at 225, 320) to the origin server if the requested object is not available on the parent servers. If the origin server has the requested object (at 325), the origin server would serve (at 230, 345) the object directly to the client (not shown in FIG. 1). Otherwise if the object is unavailable (at 335), an error message would be returned (at 340) regarding the unavailability of the requested object.

[0032] Referring to FIG. 3(b), when an edge and/or parent server determines (at 350) that a requested object is popular (by some measure of popularity) but the edge and/or parent server does not have a copy of the object, the edge and/or parent server initiates a pull of the object to the edge and/or parent server. So, for example, when the edge server determines (at 350) that a requested object is popular but the edge server does not have a copy of the requested object, the edge server initiates the replicating (at 220, 360) of the popular requested object to the edge server from a parent server that has the requested object. Similarly, for example, when a parent server 120 determines (at 350) that a requested object is popular but the parent server does not have a copy of the requested object, the parent server initiates the replicating (at 240, 360) of the popular requested object to the parent server from an origin server that has the requested object.

Alternatively, a parent and/or origin server may receive informationregarding object popularity, such as popularity determinations for objects or data about object popularity, from one or more edge and/or parent servers and may push popular objects to the edge and/or parent servers. So, for example, when the parent server determines (at 350) that a requested object is popular at an edge server but the edge server does not have a copy of the requested object, the parent server may initiate the replicating (at 220, 360) of the popular requested object to the edge server from the parent server. Similarly, for example, when the origin server determines (at 350) that a requested object is popular at a parent server but the parent server does not have a copy of the requested object, the origin server initiates the replicating (at 240, 360) of the popular requested object to the parent server from the origin server.

[0033] In some embodiments, if none of the parent servers has the requested object, the edge server initiates the replication (at 235, 360) of the popular requested object to the edge server from the origin server having the requested object (if the network has access to the origin server). Preferably, in each case, the replicated object is not served or further replicated until the object has been completely copied to the respective server. Optionally, such replicating may be utilized by and between the parent servers themselves to facilitate the reduction of the traffic to and from the origin server. Further, if the edge and/or parent server does not have adequate space for the popular requested object, one or more objects may be purged (at 355) from the edge and/or parent server to make space for the popular object. In embodiments, the method depicted in FIG. 3(c) and discussed in more detail below is employed to determine whether any object(s) in the edge and/or parent server is no longer popular and if so, to delete the no longer popular object(s) from the edge and/or parent server. Also, as will apparent to those skilled in the art, servers other than the edge and/or parent server for which an object is determined popular may perform the actual determination of whether an object is popular by using for example, popularity information provided by the handling edge and/or parent server. The popularity determinations can then be used to initiate replication (for example, pushing or pulling) of the object to the edge and/or parent server for the which the object is determined popular.

[0034] Referring to FIG. 3(c), if an object in a server's storage is no longer popular (at 365), the server may delete the object (at 370) from the storage. For example, an edge server may delete (at 245, 370) any objects from the edge server's storage that are no longer popular. Similarly, a parent server may delete (at 250, 370) any objects from the parent server's storage that are no longer popular. As will be apparent to those skilled in the art, the determining of whether any object(s) in the server's storage is no longer popular and if so, deleting the no longer popular object(s) from the server's storage may be performed independently of, for example in parallel with or before, one or more functions of the method of managed object replication and delivery. In embodiments, the no longer popular objects are removed from edge servers and, if the no longer popular objects are hosted on an origin server, from parent servers.

<u>Determining Popularity</u>

[0035] Any number of techniques may be used to determine the popularity of an object. Determining the popularity can be based on the number of requests. Popularity can also be based on the request rate. Popular objects typically have higher request rates or higher number of requests than unpopular objects. Popularity can also be determined by tracking the last X number of request times for an object and then use the difference between the current time and these request times to calculate a running average for how often the object is requested. Determining the popularity can also be gauged on the request rate for an object that is perhaps weighted for more recent requests for the object (which is a predictor that the object will be requested again). An exponential decay method and an artificial neural network could also be used to determine popularity of an object.

[0036] According to some embodiments of a popularity computation and referring to FIG. 4, the popularity of an object is based on the request rate of the object and computed over a sliding time window in a discrete manner. In these embodiments, the variable I denotes the time interval over which the popularity of an object is measured. The time interval is divided into N equal sub-intervals of duration I/N. As will be apparent, the time interval is not required to be equally divided and may instead be divided in other manners.

[0037] A linked list P of size N is created for each object. The value of N determines the quality of approximation. The smaller the value of N, the coarser the approximation. In some embodiments, the value of N is set to 5.

[0038] The first element P[1] of the list records the number of requests that arrived when the current time was within the first sub-interval, the second element P[2] records the number of requests that arrived when the current time was within the 2nd interval, and so on. When a new sub-interval arrives, the list is rotated such that P[I] becomes P[I+1] except for P[N] which becomes P[1], so, e.g., P[1] becomes P[2], P[2] becomes P[3], and P[N] becomes P[1]. After the rotation, the new P[1] is reset to zero. Accordingly, only the end time of the first sub-interval needs to be recorded and compared against the current time to check if the list should be rotated. For each new request within the sub-interval, P[1] is simply incremented by 1. In this way, the arrival time of each request need not be recorded.

[0039] In preferred embodiments, the popularity of an object is simply the sum of all numbers in the list. To make the computation more efficient, the sum of P[2] + P[3] + ... + P[N] is stored in a register M. The popularity can be then computed by adding P[1] to M. When a rotation occurs, the new value of M becomes M += P[1] - P[N]. The popularity of an object may be queried constantly. So, to avoid the extra addition involved for each such inquiry, the value of P[1] can be set to M after the rotation. Then, the value of P[1] is the popularity of the object.

[0040] The popularity computation algorithm may be summarized as follows. The linked list P of size N for an object, wherein each of P[1] . . . P[N] represents a time sub-interval, is initialized (at 400). The popularity M is also initialized (at 410). If there is a request for the object while the current time is within the current time sub-interval (at 420), then the value of P[1] is incremented (at 430) by 1. If the current time is within a new time sub-interval (at 440), then the value of P[1] is decremented by the value of M, M += P[1] - P[N], the list P is rotated and P[1] is set to the value of M (at 450). Then, provided the popularity computation is continued (at 460) e.g., the popularity computation is not terminated, the popularity computation algorithm repeats itself.

Initiating Replication

[0041] Furthermore, any number of techniques may be used to initiate replication of an object. An edge server and/or a parent server might replicate an object on the first request by a client for the object. Alternatively, the edge server and/or parent server may be tuned to wait until the edge server and/or parent server receives a specific number or range of requests for the object. In other implementations, the object may be pulled if the object is more popular (e.g., a higher request rate) than the least popular object currently in the storage. In yet another alternative, the replicating decision can be a function of the popularity of the object, the cost of storing the object, the cost of pulling the object from the network and any other relevant cost factors. However, the popularity of objects may change significantly with time. Initiating a pull decision of an object purely based on a fixed threshold does not capture this dynamic nature of popularity.

[0042] A replication policy that compares against the least popularity of replicated objects has its limitations, although the policy does not use a fixed threshold. Consider where the storage is only half full but all the replicated objects are extremely popular. Since only objects exceeding the least popularity of the replicated objects will be replicated under this replication policy, objects with moderate popularity will be rejected despite that there is plenty of storage space available and that the objects are reasonably popular.

[0043] Accordingly, a replication scheme should be able to automatically adjust the replication threshold by taking into consideration the dynamic nature of popularity and the fullness of the storage. If there are more popular objects than the storage capacity allows, the replication scheme should raise the threshold. If there is more available storage capacity, the replication scheme should decrease the threshold so that more objects can be stored.

[0044] According to embodiments of a replication scheme and referring to FIG. 5, an object is replicated (at 520) into storage when the popularity P of the object is greater (at 500) than the initial threshold P_I and when there is enough space (at 510) in the storage to replicate the object. If there is not enough storage to replicate the requested object, a replacement algorithm is performed in which the popularity P of the object is compared (at 530) against the popularity P_L of the least popular object in the storage. If P is greater than P_L, the current least popular object is removed (at 540) from the storage to free up more storage space, the next least popular object is identified (at 540), the value of the least popularity is updated (at 550), and a new iteration begins by checking if there is enough storage space to store the requested object (at 510). The storage space freeing iteration is terminated when either 1) enough storage space has been freed up to accommodate the requested object or 2) the requested object is not as popular as the least popular object in the storage. In embodiments, the least popular objects are removed from edge servers and, if there are origin servers with a copy of the least popular objects, from parent servers. Where no origin servers exist with a copy of the least popular objects, least popular objects are not removed from parent servers in order to keep a copy of the least popular objects in the network.

Purging

[0045] In some embodiments, the managed object replication and delivery method and system records the time on which an object was last requested. A purge scheme is invoked to clean up the storage of servers, for example, on a regular time interval basis or when a popular object is replicated to an edge and/or parent server but there is inadequate space at the edge and/or parent server. Referring to FIG. 6, in the purge scheme, all stale objects are removed from the storage (at 600), the remaining objects are sorted based on popularity (at 610), and the new values of P_L and P_I are determined (at 620, 630). An object is stale if its age (that is the time since the object was last requested) is over a predefined value, typically set to the duration of the sliding window used to measure the popularity multiplied by an adjustable factor. As will be apparent to those skilled in the art, the value may vary and indeed other staleness algorithms may be used. The popularity of the least popular object in the storage after purging is assigned as the new P_L. The new P_I is determined by using the sorted popularity and is set to the popularity of the last object that can fit into the storage if more popular objects are replicated first. Typically, P_L should be greater than or equal to P_I. If not, the value of P_L is assigned to be the new P_I. In some embodiments, the purge process is implemented as a separate thread in a multi-thread system. In embodiments, the stale objects are removed from edge servers and, if there are origin servers with a copy of the stale objects, from parent servers. Where no origin servers exist with a copy of the stale objects, stale objects are not removed from parent servers in order to keep a copy of the stale objects in the network.

[0046] At the outset when the system starts and there is no popularity data available yet, the initial values of both P_L and P_I can be set to zero. This forces the replication scheme to store the objects on their first request, but the purge scheme that is run on a regular basis will adjust the values of P_L and P_I automatically. The initial values of P_L and P_I can also be set to other values. Indeed, the initial values of P_L and P_I can be determined by taking into consideration the cost of storage, the cost of fetching, and the cost difference in deliveries from different servers. In any case, the system allows the specification of minimum P_L and P_I. If a computed P_L or P_I is smaller than the minimum specification, P_L or P_I is set to the minimum specification.

[0047] In some embodiments, to avoid or minimize stream thinning and other quality problems, storage fill is separated from data delivery. In this way, the data transfer between multiple sets of storages can tolerate a slower connection, and a server never streams an object unless the object is entirely in the storage. As will be apparent to those skilled in the art, it is possible to start streaming an object when there is enough data in the storage and that replication need not be completed before serving the object. Further, storage fill may be staged by copying to a separate location, then moving the copied data to a servable location when the data is complete.

[0048] Further, if an object is changed at an origin server, there may be a need to broadcast a message to remove the object at one or more parent servers and/or one or more edge servers. Similarly, if an object is changed at the parent server(s), there may be a need to broadcast a message to remove the object at one or more edge servers. In each case, future requests for the removed object would be handled as in the normal case where a requested object is not available at an edge server and/or a parent server.

Hardware and Software

[0049] In embodiments of the invention, referring to FIGS. 1 and 7, the system of managed object replication and delivery comprises one or more servers in a network designated as parent servers and one or more servers in the network designated as edge servers. In some embodiments, referring to FIG. 1, parent servers 120 have large storage capacity (on the order of 5 terabytes (TB)) while edge servers 130 have smaller storage space (ranging from 1 TB to 500 GB). One or more redirectors for implementing the method of managed object replication and delivery are installed on each edge server cluster. In some embodiments, one or more objects are replicated to one or more of the parent servers from the origin servers and then pulled from the parent servers to the edge servers as needed. In other embodiments, one or more objects are replicated to one or more of the edge servers and/or to one or more of the parent servers, from the origin servers as needed.

[0050] In some embodiments, a data transfer method 700, 710 is implemented to transfer data between parent servers and edge servers. The data transfer method supports the Transport Layer Security (TLS) protocol (described in the Internet Engineering Task Force (IETF) RFC 2246, located at "http://www.ietf.org/rfc/rfc2246.txt", incorporated by reference herein) to ensure communication privacy. Further, the implementation of the method for managed object replication and delivery supports three popular object formats, namely Apple Computer, Inc.'s QuickTime^{TM}, RealNetworks, Inc.'s Real^{TM}, and Microsoft Corporation's WindowsMedia^{TM} formats for streaming of requested object(s). As will be apparent to those skilled in the art, any number of other protocols and object formats may be used.

[0051] Further, in some embodiments, a number of software components are used to facilitate the method of managed object replication and delivery. A first component is a WindowsMedia redirector 720, 760 which is a service running on the Microsoft Windows NT operating system that processes requests from a Windows Media player and performs the redirection of the request for Windows Media objects. The WindowsMedia redirector is provided on edge servers and parent servers. Currently, the Microsoft Media Server (MMS) protocol is used for streaming of Windows Media objects and that protocol does not support redirection. To provide redirection for the streaming of Windows Media objects, the uniform resource identifier (URI) hyperlinks at the customer's site for such streaming Windows Media objects are modified. URIs as used herein generally have the following form (defined in detail in T. Berners-Lee et al, *Uniform Resource Identifiers*

(URI), IETF RFC 2396, August 1998, located at "http://www.ietf.org/rfc/rfc2396.txt", incorporated by reference herein):

```
scheme://host[port]/uri-path
```

where "scheme" can be a symbol such as "http" (see *Hypertext Transfer Protocol – HTTP/1.1*, IETF RFC 2616, located at "http://www.ietf.org/rfc/rfc2616.txt", incorporated by reference herein) for an object on a Web server or "rtsp" (see *Real Time Streaming Protocol (RTSP),* IETF RFC 2326, located at "http://www.ietf.org/rfc/rfc2326.txt", incorporated by reference herein) for an object on a streaming server. Other schemes can also be used and new schemes may be added in the future. The port number "port" is optional, the system substituting a default port number (depending on the scheme) if none is provided. The "host" field maps to a particular network address for a particular computer. The "uri-path" is relative to the computer specified in the "host" field. An uri-path is typically, but not necessarily, the path-name of a file in a media server directory. In a preferred embodiment, the HTTP protocol is used to effect the redirection of WindowsMedia objects. Therefore, the "scheme" field of the URIs of the WindowsMedia objects is changed from "mms" to "http". For example, the URI for a sample object "sample.asf" in the Windows Media Advanced Streaming Format (ASF) will have a new URI of the form "http://host/path/sample.asf". For objects using Windows Media ASX scripting, a sample URI for the "meta.asx" object will be in the form "http://host/?www.customer.com/path/meta.asx", where "customer" is the name of the content provider of "meta.asx". All URIs contained within the "meta.asx" object remain unchanged. Upon receiving the request "http://host/path/sample.asf", the WindowsMedia redirector would respond to the request with the following example ASX script:

```
<ASX version = "3.0">
<Entry><Ref href= "mms://servername/path/sample.asf" /></Entry>
</ASX>
```

in the message body, if the requested object is found available either locally or on another server (parent or origin). In this example, "servername" is or resolves to the Internet Protocol (IP) address of a media server that will serve the requested object to the requesting client. If the requested object cannot be found, the WindowsMedia redirector would respond to the request with the following example ASX script:

```
<ASX version = "3.0">
<Entry><Ref href= "http://redirname/path/sample.asf" /></Entry>
</ASX>
```

in the message body, where "redirname" is or resolves to the IP address of the redirector of a parent server, to trigger another round of redirection. A final round of redirection is reached when none of the parent servers (and the origin server, if applicable) has the requested object. In this case, the redirection process is terminated, and a "not found" error message is sent to the requesting client. Requests for ASX objects are processed in a similar way. Upon receiving the request for the sample object "meta.asx", the WindowsMedia redirector checks the availability of the object pointed to by each URI inside "meta.asx" and rewrites the URI of each object accordingly. Then the WindowsMedia redirector sends a response to the request with the rewritten "meta.asx" in the message body of the response. The URI rewriting is done as follows. If a requested object, for example, "file.asf", is found available locally or on another server, the corresponding URI would be rewritten to "mms://servername/path/file.asf", where "servername" is or resolves to the IP address of the media server that will serve the requested object to the requesting client. If "file.asf" cannot be found, the corresponding

URI is rewritten to “http://redirectorname/path/file.asf”, where “redirname” is or resolves to the IP address of a parent server redirector.

[0052] Another component is a Real/QuickTime redirector 730, 770 which is an application that processes Real-Time Streaming Protocol (RTSP) requests from a Real or QuickTime player for one or more objects and performs the redirection of the method for Real and QuickTime objects. The Real/QuickTime redirector is provided on edge servers and parent servers. The RTSP, described in detail in the IETF RFC 2326, is used for streaming Real and QuickTime objects, and the “REDIRECT” method supported in that protocol is used to effect redirection. A redirect request informs the client that it must reconnect to another server location and provides for the client the URI of that new server in the redirect request.

[0053] A best or optimal server selection mechanism is also provided (not shown in FIG. 7). The best or optimal server selection mechanism includes selection of an edge server most suitable for delivery of one or more objects to the client according to any number of currently known or future developed algorithms. In addition to redirection to a best or optimal edge server for handling a client request for an object, the best or optimal server mechanism may also be applied to trigger one or more further redirections to one or more parent server(s) when a requested object is not available at the handling edge server. In an implementation, to effect this operation, the hostname part of the URI for a requested object is modified. For example, in the link “http://customer-wm.fpondemand.net/customer/sample.asf”, “customer-wm.fpondemand.net” would be changed to “parent-wm.fpondemand.net” forcing the request to go through a further round of best or optimal server selection against parent servers only. In such embodiments, to effect best or optimal parent server selection, the parent-edge server topology is defined and the best or optimal server selection mechanism is provided a parent server table defining the relationships of such a topology. In some embodiments, the best or optimal server selection mechanism is similar to the best repeater selector described in U.S. Patent No. 6,185,598.

[0054] A file replication manager application 740, 750 is also provided that manages object replication to and object removal from storage, retrieves objects from parent servers for replication to edge server storage, and performs storage cleanup as needed. The file replication manager is provided on edge servers and parent servers. In some embodiments, the file replication manager application uses the data transfer method and is in communication with the WindowsMedia and Real/QuickTime redirectors to provide, if available in the storage, objects requested by those redirectors.

[0055] In some embodiments, the message communicated between a WindowsMedia or a Real/QuickTime redirector and a file replication manager and between file replication managers is encapsulated using the User Datagram Protocol (UDP). This allows address handling and delivery to be handled by UDP and facilitates fast communication. Since UDP does not guarantee delivery, the message header contains a message number to be used to confirm that a response is to the current query, and not to a previous query. In addition, MD5 (See, e.g., Rivest, R., "The MD5 Message Digest Algorithm", IETF RFC 1321, April 1992) is supported to provide a basic level of security. The MD5 hash is generated by running a MD5 hash algorithm on the message number, message, and a secret pass phrase only shared by components of the system of managed object replication and delivery. When a message is received, the MD5 hash of the message number, message, and secret pass phrase, is computed and compared against the MD5 hash provided in the message. If these two MD5 hashes do not match, the message is invalid, and will be discarded.

[0056] As will be apparent to those skilled in the art, FIG. 7 represents only some embodiments of the system according to the present invention. Many variations for implementing the system according to the teachings of the present invention are possible and are all within the scope of the invention.

Chunking

[0057] An extension of the above method and system is to provide chunking. Studies of log data show that, even for popular objects, a good percentage of requests for such objects exit before the object is completely served. To exploit this kind of object usage and further enhance the performance of the network, objects can be segmented into chunks and initial chunks of an object can be given preferential treatment in the replication scheme. For example, only the initial chunks of a object are replicated when a replication admission decision is made and the remaining chunks of the object are pulled to the storage only if the client does not exit before a certain amount or number (e.g., 90%) of the initial chunks of the object are served. The initial chunks of an object can be left in the storage even when the object becomes unpopular. By partitioning streams in this manner, a first part of an object can be served from edge servers quickly, even if most of the object stream must be fetched from a parent server or origin server.

Object Retention and Staleness

[0058] Optionally, some or all of the objects may be permanently retained in edge server storage or be retained depending on a quota. Similarly, a configurable or automatically adjusting threshold for storage filling and deletion may be provided.

[0059] Also, an edge server may be configured to determine whether a requested object in a server's storage is fresh and serve the requested object only when the object is not stale. In some embodiments, a file is maintained which lists the maximum storage age and storage quota in order to facilitate determining whether a requested object is fresh. If a request is received for a stale object a redirect is initiated to the relevant parent server or origin server to provide the requested object and a storage refresh will be performed if the requested object is popular.

Peers

[0060] Also, edge server storage fills of objects may be served by other peer edge servers instead of a relevant parent server or origin server. If a popular object has already been replicated to an edge server filling a new edge server request for that object from one of the peer edge servers may be more efficient than the parent server or origin server. Since there are typically more edge servers than parent servers and origin servers, there is an increased likelihood that a peer edge server may be closer in terms of network distance than a relevant parent server or origin server. Moreover, such peer edge server storage fills could also lessen the burden on the parent servers or origin servers.

[0061] The detailed descriptions may have been presented in terms of program procedures executed on a computer or network of computers. These procedural descriptions and representations are the means used by those skilled in the art to most effectively convey the substance of their work to others skilled in the art. The embodiments of the invention may be implemented as apparent to those skilled in the art in hardware or software, or any combination thereof. The actual software code or hardware used to implement the invention is not limiting of the invention. Thus, the operation and behavior of the embodiments often will be described without specific reference to the actual software code or hardware components. The absence of such specific references is feasible because it is clearly understood that artisans of ordinary skill would be able to design software and hardware to implement the embodiments of the invention based on the description herein with only a reasonable effort and without undue experimentation.

[0062] A procedure is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. These operations comprise physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, objects, attributes or the like. It should be noted, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

[0063] Further, the manipulations performed are often referred to in terms, such as adding or comparing, which are commonly associated with mental operations performed by a human operator. No such capability of a human operator is necessary, or desirable in most cases, in any of the operations of the invention described herein; the operations are machine operations. Useful machines for performing the operations of the invention include general purpose digital computers, special purpose computers or similar devices.

[0064] Each operation of the method may be executed on any general computer, such as a mainframe computer, personal computer or the like and pursuant to one or more, or a part of one or more, program modules or objects generated from any programming language, such as C++, Perl, Java^{TM}, Fortran, etc. And still further, each operation, or a file, module, object or the like implementing each operation, may be executed by special purpose hardware or a circuit module designed for that purpose. For example, the invention may be implemented as a firmware program loaded into non-volatile storage or a software program loaded from or into a data storage medium as machine-readable code, such code being instructions executable by an array of logic elements such as a processor or other digital signal processing unit. Any data handled in such processing or created as a result of such processing can be stored in any memory as is conventional in the art. By way of example, such data may be stored in a temporary memory, such as in the RAM of a given computer system or subsystem. In addition, or in the alternative, such data may be stored in longer-term storage devices, for example, magnetic disks, rewritable optical disks, and so on.

[0065] In the case of diagrams depicted herein, they are provided by way of example. There may be variations to these diagrams or the operations described herein without departing from the spirit of the invention. For instance, in certain cases, the operations may be performed in differing order, or operations may be added, deleted or modified.

[0066] Embodiments of the invention may be implemented as an article of manufacture comprising a computer usable medium having computer readable program code means therein for executing the method operations of the invention, a program storage device readable by a machine, tangibly embodying a program of instructions executable by a machine to perform the method operations of the invention, or a computer program product. Such an article of manufacture, program storage device or computer program product may include, but is not limited to, CD-ROM, CD-R, CD-RW, diskettes, tapes, hard drives, computer system memory (e.g., RAM or ROM), and/or the electronic, magnetic, optical, biological or other similar embodiments of the program (including, but not limited to, a carrier wave modulated, or otherwise manipulated, to convey instructions that can be read, demodulated/decoded and executed by a computer). Indeed, the article of manufacture, program storage device or computer program product may include any solid or fluid transmission medium, whether magnetic, biological, optical, or the like, for storing or transmitting signals readable by a machine for controlling the operation of a general or special purpose computer according to any or all methods of the invention and/or to structure its components in accordance with a system of the invention.

[0067] Embodiments of the invention may also be implemented in a system. A system may comprise a computer that includes a processor and a memory device and optionally, a storage device, an output device such as a video display and/or an input device such as a keyboard or computer mouse. Moreover, a system may comprise an interconnected network of computers. Computers may equally be in stand-alone form (such as the traditional desktop personal computer) or integrated into another apparatus (such as a cellular telephone).

[0068] The system may be specially constructed for the required purposes to perform, for example, the method of the invention or the system may comprise one or more general purpose computers as selectively activated or reconfigured by a computer program in accordance with the teachings herein stored in the computer(s). The system could also be implemented in whole or in part as a hard-wired circuit or as a circuit configuration fabricated into an application-specific integrated circuit. The invention presented herein is not inherently related to a particular computer system or other apparatus. The required structure for a variety of these systems will appear from the description given.

[0069] While this invention has been described in relation to certain embodiments, it will be understood by those skilled in the art that other embodiments according to the generic principles disclosed herein, modifications to the disclosed embodiments and changes in the details of construction, arrangement of parts, compositions, processes, structures and materials selection all may be made without departing from the spirit and scope of the invention Changes, including equivalent structures, acts, materials, etc., may be made, within the purview of the appended claims, without departing from the scope and spirit of the invention in its aspects. Thus, it should be understood that the above described embodiments have been provided by way of example rather than as a limitation of the invention and that the specification and drawing(s) are, accordingly, to be regarded in an illustrative rather than a restrictive sense. As such, the invention is not intended to be limited to the embodiments shown above but rather is to be accorded the widest scope consistent with the principles and novel features disclosed in any fashion herein.